

Machine Learning 1

Lecture 10.4 - Unsupervised Learning

Non-linear PCA

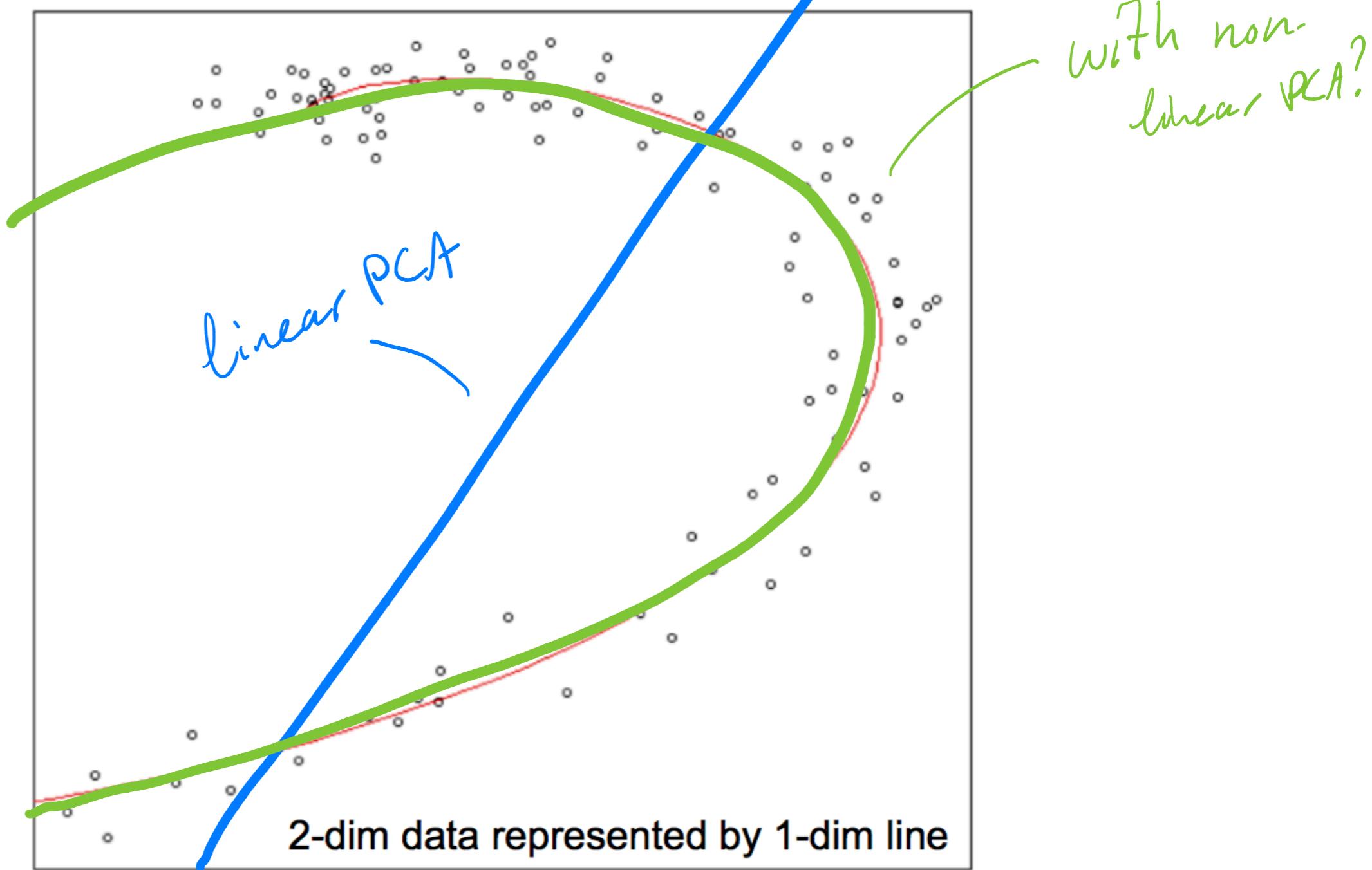
Erik Bekkers

(Bishop 12.3, (12.4.1), 12.4.2)



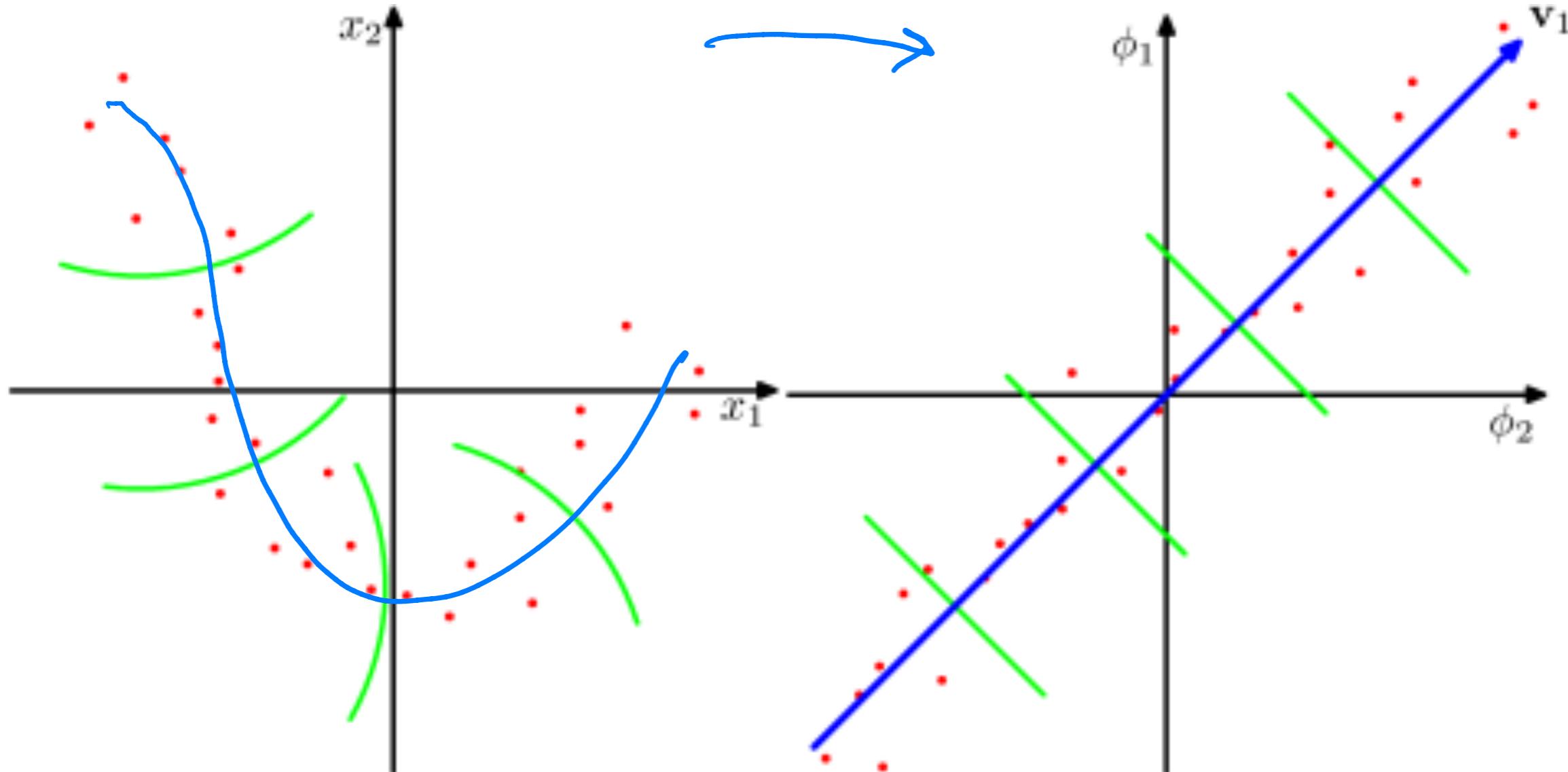
PCA on a spiral

Dimensionality reduction



PCA using basis functions

use predefined feature vectors $\phi(x)$



(Bishop 12.16)

How to choose ϕ ?

Kernel PCA

(Bishop 12.3)

$$\text{Cov}[\mathbf{x}] = \frac{1}{N} \sum_n (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$$

- Use feature transformations to “linearize” the data

- Via some specific choice of basis functions $\phi(\mathbf{x}_n)$
- Do PCA in this space

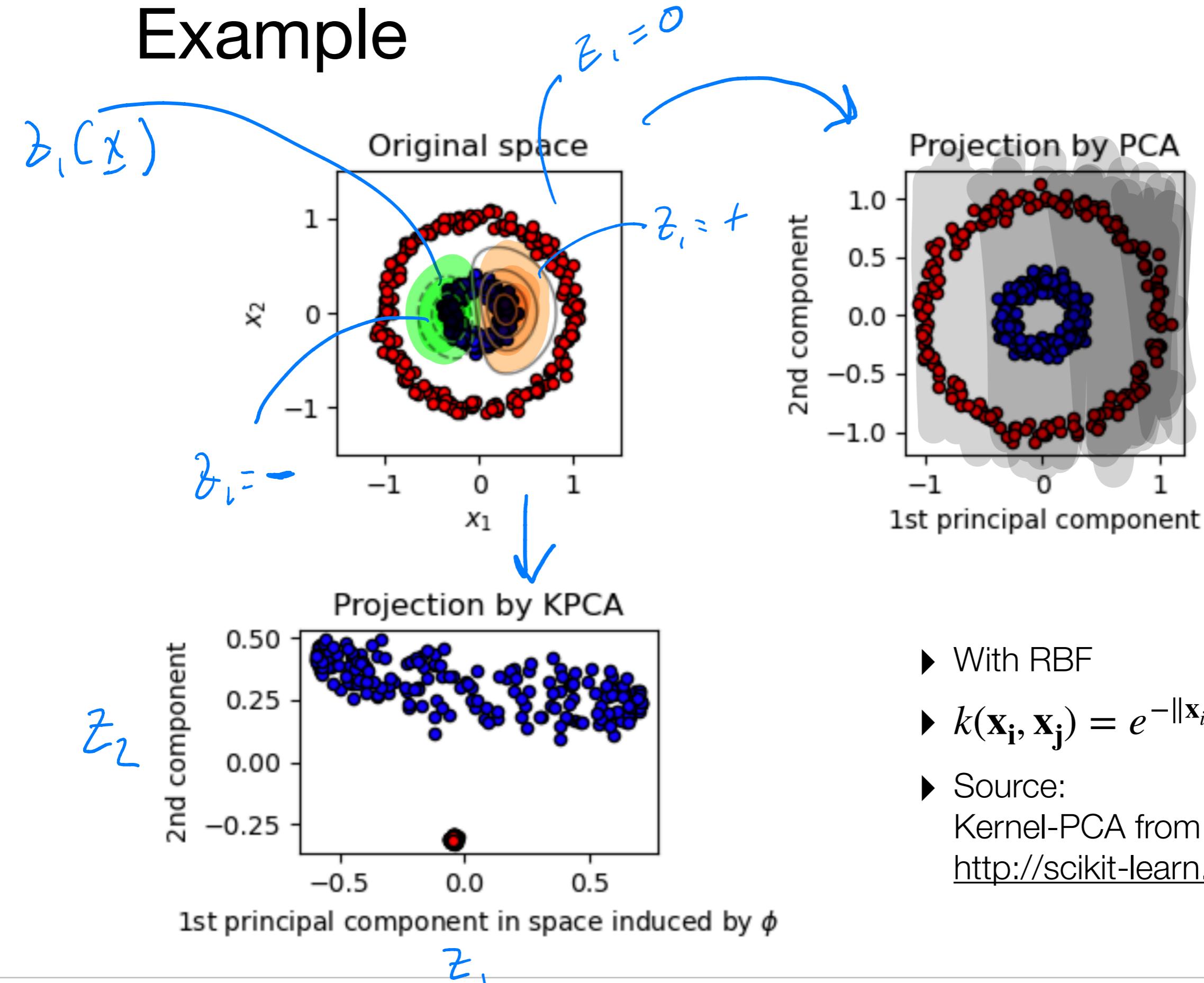
$$S = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T,$$

$$C = \frac{1}{N} \sum_{n=1}^N \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T$$

- Kernel approach:

- Let $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ be the kernel associated with the basis functions
- Let \mathbf{u}_i be the principal components of $C \in \mathbb{R}^{M \times M}$
- Let $z_i(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{u}_i$ the projection onto the i^{th} component
- Let \mathbf{a}_i be the i^{th} eigenvector of $\mathbf{K} = \Phi^T \Phi \in \mathbb{R}^{N \times N}$
- Then $z_i(\mathbf{x}) = \sum_{n=1}^N a_{in} k(\mathbf{x}, \mathbf{x}_n)$ - The projection is purely in terms of the other data points via k !
- Kernel trick: use some defined kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ without explicitly defining the basis functions ϕ
 Powerful! : implicitly work with infinite dimensional features M=∞

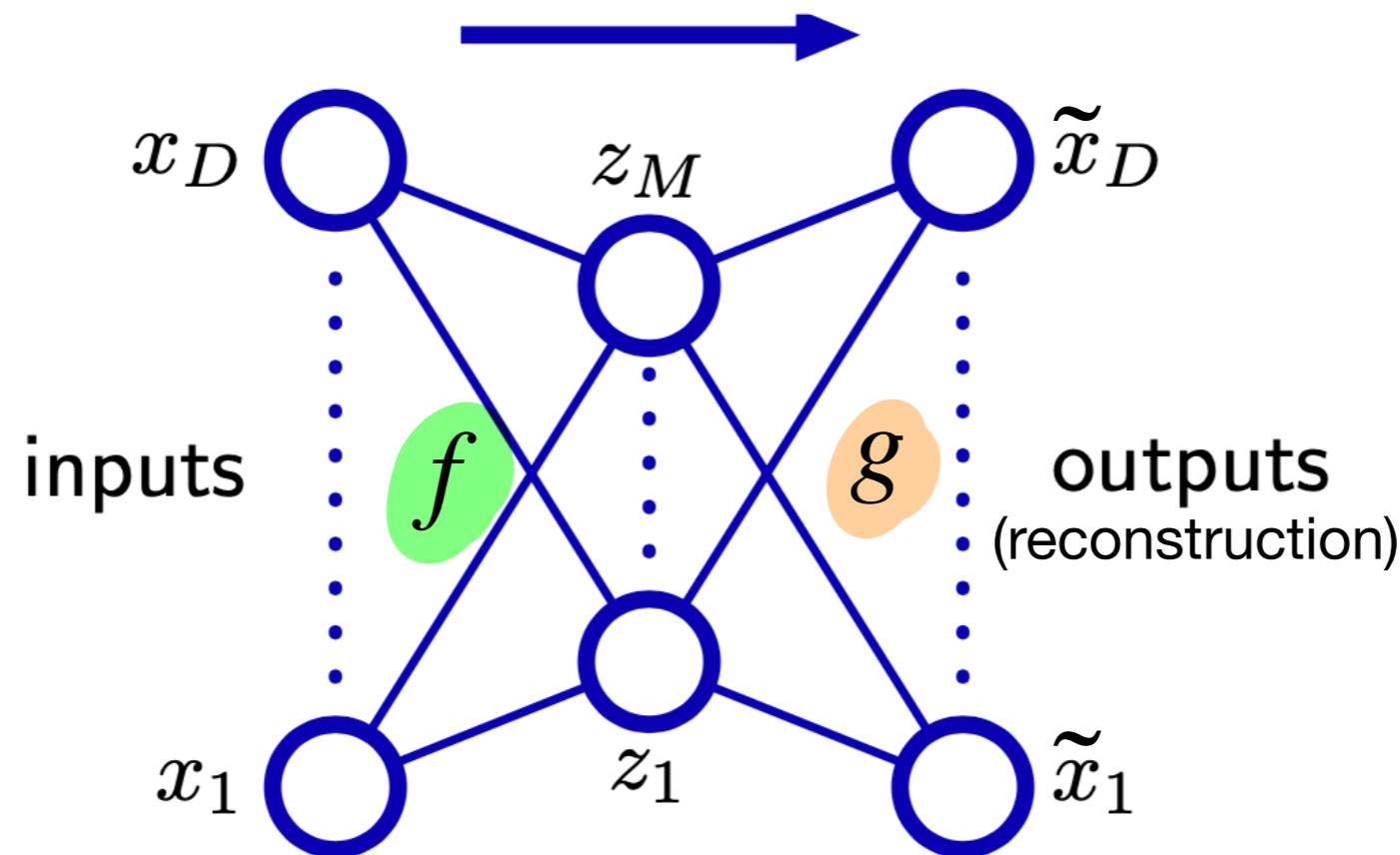
Example



- With RBF
- $k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}$
- Source:
Kernel-PCA from
<http://scikit-learn.org>

Auto-encoders (auto-associative neural nets)

- Non-linear dimensionality reduction with **neural networks**
- Maps:
 - Encoder (projection): $\mathbf{z} = f(\mathbf{x})$
 - Decoder (reconstruction): $\tilde{\mathbf{x}} = g(\mathbf{z})$
 - Both are neural networks
- Goal: minimize the reconstruction error



(Bishop 12.18)

Autoencoder objective

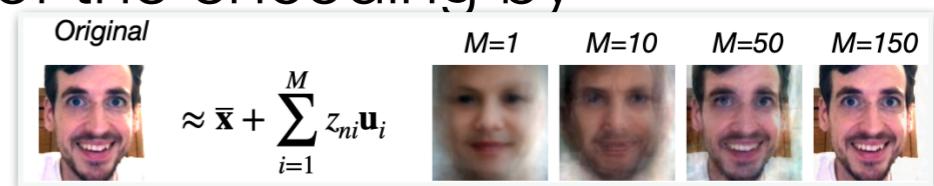
- Minimize the error between original and reconstructed input:

$$\frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - g_{\mathbf{w}'}(\underbrace{f_{\mathbf{w}}(\mathbf{x}_n)}_{z})\|^2$$

- Non-linear, no closed form solution, solve via SGD!
- Recall minimum error viewpoint of PCA:
$$f(\mathbf{x}) = \mathbf{U}_{\mathbf{M}}^T (\mathbf{x} - \bar{\mathbf{x}}), \quad g(\mathbf{x}) = U_{\mathbf{M}} \mathbf{z} + \bar{\mathbf{x}}$$
- PCA \leftrightarrow 2 layer autoencoder without activation functions

Autoencoder as generator

- As before, we could judge visually the quality of the encoding by looking at reconstructed images
- Bonus: we can use $\tilde{\mathbf{x}} = g(\mathbf{z})$ as a **generator of fake data** (images):
 - Train the autoencoder to get $\tilde{\mathbf{x}} = g(\mathbf{z})$
 - Sample a random $\mathbf{z}^* \sim p(\mathbf{z})$
 - Feed it into the generator to get fake image $\tilde{x} = g(\mathbf{z})$



**A probabilistic model needs to be defined before sampling, this definition needs to be part of the encoding/decoding optimization pipeline, see Variational Autoencoders (VAE) [Kingma & Welling, ICLR 2014]*

Autoencoder on MNIST

Which one is real?



... autoencoders in 2018



Trained on CelebA in [Patrini et al., Sinkhorn AutoEncoders, 2018]